Encapsulation Solutions for Year 2000 Compliance:
A Summary

Don Estes
2000 Technologies Corporation

This article discusses two variations of time-shifting strategies for year 2000 compliance: data encapsulation and program encapsulation. A summary of the complete article follows; the detailed article, containing specific strategies and examples, can be found on the CROSSTALK Web site at http://www.stsc.hill.af.mil/CrossTalk/crostalk.html.

Standard technical strategies to achieve year 2000 (Y2K) compliance include replacement, date expansion, and various forms of windowing. Recently added to this list are time-shifting strategies, which significantly reduce costs, business risks, and time to implement and test. All of the benefits of encapsulation result from two central facts:

- Analysis and implementation efforts are minimal.
- It is the inverse of the procedure used to age data into the future to establish future data compliance; once one successfully proves current-year regression testing, there is an implicit establishment of future-dated regression testing for the length of time constant employed.

A different implementation of the encapsulation concept can be employed as a test harness to perform automated Y2K testing, including dynamic future date data aging. A description of the automated testing implementation is planned for a future *CROSSTALK* article.

Time-shifting strategies dramatically reduce the costs of testing by eliminating the need to build future-dated test cases, and because they use a complementary automated regression testing facility, they are bound into the program logic with the encapsulation logic. The risk profile also is minimal because the relatively small amount of affected program code is at the boundary between the program and the outside data storage. Because the file formats do not change, it also has minimal implementation and deployment impact.

The automation of date expansion and procedural logic solutions offer many of the implementation advantages of encapsulation, including cost reduction. Limited windowing methods can compete with encapsulation on an implementation cost basis for many applications. However, in the area of testing, encapsulation is in a class by itself. Only encapsulation can bypass the expense of future-dated testing altogether and, through automation, bypass the manual construction of unit test data. As the event horizon draws near, this aspect of encapsulation will dominate strategy decisions.

Data Encapsulation vs. Program Encapsulation

The difference between data and program encapsulation is in what you change. The mnemonic rule is you encapsulate what you do not change. So, if you are changing programs but not data, you are performing data encapsulation. But if you are changing data but not programs, you are performing program encapsulation. A hybrid of the two interposes a layer between program and data to perform the time shifting.

Encapsulation strategies are similar to windowing strategies in that a two-digit year is maintained. However, procedural logic strategies infer the century from the data and operate while spanning the century boundary. Time-shifting strategies, by contrast, shift the data back in time to avoid the century boundary altogether. The essential problem with maintaining a two-digit year is that 2000 > 1999 but 00 < 99. By shifting the dates back in time, typically by a multiple of 28 years, we end up with 1972 > 1971 and 72 > 71, which solves the problem. As long as all dates are shifted consistently, we receive the same results for the same input, and the applications will work until 2027 or 2055. Once all stored data are in the 21st century, the time shift can be turned off, at which time the application will work until 2100. This can be considered a permanent fix.

One absolute requirement with this method is that no two-digit years can be stored from before 1929 (for a 28-year shift) for any date used in a comparison or a calculation, although there are special case exceptions to this rule. The reason is that once shifted, the date data must all lie in the same century. This requirement does not apply to dates used merely for storage and retrieval.

Encapsulation is unique because the indeterminacy of assessment is eliminated; this is the major source of delay, even in windowing and expansion projects using automated assessment tools of great power. The single data entry and exit points in each program are vastly fewer in number and are essentially decoupled from each other, so that

one need only examine those points and the data flowing through them to answer all questions required for encapsulation. If any doubts remain, one need only dump the relevant data files or tables and look at them.

Data encapsulation works on a program-by-program basis, as compared to program encapsulation, which works on a system-by-system basis. As a result, within the same system it is possible to use data encapsulation for one program, a standard windowing solution for another, and to leave a third completely unchanged because date data flows through the program without being processed in any way. This may be important for sites that plan to use encapsulation as a short-term fix while preparing a more comprehensive solution via windowing, expansion, or replacement.

Program encapsulation may have an implementation advantage over data encapsulation, although this will be significant primarily in larger projects. This is because programs that do not cross the time-warp zone boundary usually do not require modification.

Encapsulation Metrics

Data encapsulation was first proposed by a major defense contractor in 1992, and we are now aware of some two dozen pilot and full projects that use the method. In addition, both program and data encapsulation can now be automated. Early metrics show an average of 1,000 to 2,000 lines of code per day per programmer for manual data encapsulation implementation, and 10 times this for automated implementation. Program encapsulation can be even more efficient, particularly for larger projects.

Conclusion

Encapsulation, although a new strategy relative to expansion, replacement, or windowing, is rapidly proving itself as the most efficient in terms of time and cost and will increasingly be the center of consideration as we move toward our time horizon for failure. ◆

About the Author

Don Estes is chief technology officer for 2000 Technologies Corporation, for whom he has designed and implemented both a data encapsulation and an automated testing system. He also works closely with vendors of limited windowing, program encapsulation, and object code remediation systems. He has been involved with COBOL and database applications for 25 years and database and mainframe performance tuning for 10 years. For the last seven years, he has helped design and execute projects for the mass modification of large bodies of source code, primarily for platform migration, using state-of-the-art automated source language transformation technologies and automated testing methods. He is a regular contributor to Peter de Jager's Year 2000 mail list, where he is known for his contributions relating to Y2K rapid compliance strategies and automated testing. Estes is a graduate of Massachusetts Institute of Technology in physics, with a postgraduate degree from the University of Texas in educational psychology.

2000 Technologies Corporation
114 Waltham Street, Suite 19
Lexington, MA 02173
Voice: 781-860-5277, 1-800-756-8046
E-mail: info@2000technologies.com

Call for Articles

If your experience or research has produced information that could be useful to others, *CROSSTALK* will get the word out. Not only is *CROSSTALK* a forum for high-profile leaders, it is an effective medium for useful information from all levels within the Department of Defense (DoD), industry, and academia.

Published monthly, *CROSSTALK* is an official DoD periodical distributed to over 19,000 readers, plus uncounted others who are exposed to the journal in offices, libraries, the Internet, and other venues. *CROSSTALK* articles are also regularly reprinted in other publications.

We welcome articles on all software-related topics, but are especially interested in several high-interest areas. Drawing from reader survey data, we will highlight your most requested article topics as themes for 1998 *CROSSTALK* issues. In future issues, we will place a special, yet nonexclusive, focus on

Internet/Intranet
June 1998
Article Submission Deadline: Feb. 2, 1998

Project Management
July 1998
Article Submission Deadline: March 2, 1998

Measure and Metrics
August 1998
Article Submission Deadline: April 6, 1998

Look for additional announcements that reveal more of our future issues' themes. We will accept article submissions on all software-related topics at any time; our issues will not focus exclusively on the featured theme.

Please follow the *Guidelines for CROSSTALK Authors*, available on the Internet at http://www.stsc.hill.af.mil/. Hard copies of the guidelines are also available upon request. All articles must be approved by the *CROSSTALK* Editorial Board prior to publication. We do not pay for articles. Send articles to

Ogden ALC/TISE
ATTN: Heather Winward, Crosstalk Features Coordinator
7278 Fourth Street
Hill AFB, UT 84056-5205

Or E-mail articles to winwardh@software.hill.af.mil/. For additional information, call 801-777-9239.

Tracy Stauder
Managing Editor